

Ejercicios prácticos 1

ALUMNO: Juan Carlos Montes González
Jennifer Resendiz Isidro

PROFESOR: Dr. Eduardo Cornejo Velázquez



Introduccion

Se ha desarrollado una serie de ejercicios prácticos para aprender y reforzar los conceptos fundamentales de las bases de datos distribuidas con MySQL. Los ejercicios se basan en dos tablas principales: Employee y Reward, que simulan un entorno típico de gestión de empleados y compensaciones en una organización.

A través de estos ejercicios, practicamos operaciones SQL, como la creación de tablas, la inserción de datos, la ejecución de consultas de selección, la manipulación de cadenas y el uso de funciones integradas para transformar y filtrar datos. Cada ejercicio está diseñado para desarrollar habilidades de escritura de sentencias SQL eficientes y correctas, esenciales para la gestión de bases de datos relacionales y distribuidas.

Employee_id	First_name	Last_name	Salary	Joining_date	Departement
1	Bob	Kinto	1000000	2019-01-20	Finance
2	Jerry	Kansxo	6000000	2019-01-15	IT
3	Philip	Jose	8900000	2019-02-05	Banking
4	John	Abraham	2000000	2019-02-25	Insurance
5	Michael	Mathew	2200000	2019-02-28	Finance
6	Alex	chreketo	4000000	2019-05-10	IT
7	Yohan	Soso	1230000	2019-06-20	Banking

Employee_ref_id	date_reward	amount
1	2019-05-11	1000
2	2019-02-15	5000
3	2019-04-22	2000
1	2019-06-20	8000

Marco teorico

Álgebra relacional

Operaciones unarias

- Selección (σ)
- Proyección (Π)
- Renombramiento (ρ)

Operaciones binarias

- Unión (\cup)
- Diferencia de conjuntos ($-$)
- Producto cartesiano (\times)

SQL

Definición y Origen

SQL (*Structured Query Language*) es un lenguaje de programación diseñado para gestionar y manipular **bases de datos relacionales**. Fue desarrollado inicialmente por IBM en la década de 1970 bajo el nombre *SEQUEL*, y posteriormente estandarizado por ANSI (*American National Standards Institute*) e ISO (*International Organization for Standardization*).

Propósito Principal

SQL permite:

- Crear, modificar y eliminar estructuras de bases de datos (tablas, índices, vistas).
- Insertar, actualizar, eliminar y consultar datos.
- Gestionar permisos y transacciones.
- Optimizar el rendimiento de las consultas.

Componentes del SQL

Algunos componente son:

Define estructuras de datos.

Ej: CREATE, ALTER, DROP.

Manipula datos dentro de las tablas.

Ej: SELECT, INSERT, UPDATE, DELETE.

Bases de Datos Relacionales

SQL opera sobre bases de datos relacionales, que organizan los datos en **tablas** compuestas por filas (registros) y columnas (atributos). Las tablas se relacionan entre sí mediante claves primarias (PK) y claves foráneas (FK).

Ejercicios

1. Escribe la sintaxis para crear la tabla "Employee".

```
create table Employee(  
    Employee_id int not null,  
    First_name varchar(255) not null,  
    Last_name varchar(255) not null,  
    Salary int not null,  
    Joining_date varchar(255),  
    Departament varchar(255),  
    Primary key(Employee_id));
```

```
mysql> create table Employee(  
    -> Employee_id int not null,  
    -> First_name varchar(255) not null,  
    -> Last_name varchar(255) not null,  
    -> Salary int not null,  
    -> Joining_date varchar(255),  
    -> Departament varchar(255),  
    -> Primary key(Employee_id)  
    -> );  
Query OK, 0 rows affected (0.13 sec)
```

2. Escribe la sintaxis para insertar 7 registros (de la imagen) a la tabla "Employee".

Employee ← Employee E

```
insert into Employee values  
(1, 'Bob', 'Kinto', 1000000, '2019-01-20', 'Finance'),  
(2, 'Jerry', 'Kanaxo', 6000000, '2019-01-15', 'IT'),  
(3, 'Philip', 'Jose', 8900000, '2019-02-05', 'Banking'),  
(4, 'John', 'Abraham', 2000000, '2019-02-25', 'Insurance'),  
(5, 'Michael', 'Mathew', 2200000, '2019-02-28', 'Finance'),  
(6, 'Alex', 'Chreketo', 4000000, '2019-05-10', 'IT'),  
(7, 'Yohan', 'Soso', 1230000, '2019-06-20', 'Banking');
```

```
mysql> insert into Employee values (1, 'Bob', 'Kinto', 1000000, '2019-01-20', 'Finance'),  
    -> (2, 'Jerry', 'Kanaxo', 6000000, '2019-01-15', 'IT'),  
    -> (3, 'Philip', 'Jose', 8900000, '2019-02-05', 'Banking'),  
    -> (4, 'Jonh', 'Abraham', 2000000, '2019-02-25', 'Insurance'),  
    -> (5, 'Michael', 'Mathew', 2200000, '2019-02-28', 'Finance'),  
    -> (6, 'Alex', 'Chreketo', 4000000, '2019-05-10', 'IT'),  
    -> (7, 'Yohan', 'Soso', 1230000, '2019-06-20', 'Banking');  
Query OK, 7 rows affected (0.04 sec)  
Records: 7 Duplicates: 0 Warnings: 0
```

3. Escribe la sintaxis para crear la tabla "Reward".

```
create table Reward(  
    Employee_ref_id int not null,  
    date_reward varchar(255) not null,  
    amount int not null);
```

```
mysql> create table Reward(
-> Employee_ref_id varchar(255) not null,
-> date_reward varchar(255) not null,
-> amount int not null
-> );
Query OK, 0 rows affected (0.03 sec)
```

4. Escribe la sintaxis para insertar 4 registros (en la imagen) a la tabla "Reward".

σ Reward \leftarrow Reward E

```
insert into Reward values
(1, '2019-05-11', 1000),
(2, '2019-02-15', 5000),
(3, '2019-06-20', 2000),
(1, '2019-01-25', 3000);
```

```
mysql> insert into Reward values (1, '2019-05-11', 1000),
-> (2, '2019-02-15', 5000),
-> (3, '2019-04-22', 2000),
-> (4, '2019-06-20', 8000);
Query OK, 4 rows affected (0.01 sec)
Records: 4 Duplicates: 0 Warnings: 0
```

5. Obtener todos los empleados.

σ Employee

```
select * from Employee;
```

```
mysql> select * from Employee;
+-----+-----+-----+-----+-----+-----+
| Employee_id | First_name | Last_name | Salary | Joining_date | Departament |
+-----+-----+-----+-----+-----+-----+
|          1 | Bob       | Kinto    | 1000000 | 2019-01-20   | Finance     |
|          2 | Jerry     | Kanaxo   | 6000000 | 2019-01-15   | IT          |
|          3 | Philip    | Jose     | 8900000 | 2019-02-05   | Banking     |
|          4 | Jonh      | Abraham  | 2000000 | 2019-02-25   | Insurance   |
|          5 | Michael   | Mathew   | 2200000 | 2019-02-28   | Finance     |
|          6 | Alex      | Chreketo | 4000000 | 2019-05-10   | IT          |
|          7 | Yohan     | Soso     | 1230000 | 2019-06-20   | Banking     |
+-----+-----+-----+-----+-----+-----+
7 rows in set (0.01 sec)
```

6. Obtener el primer nombre y apellido de todos los empleados.

$\Pi_{First_name, Last_name}(Employee)$

```
select first_name, last_name from Employee;
```

```
mysql> select first_name, last_name from employee;
+-----+-----+
| first_name | last_name |
+-----+-----+
| Bob       | Kinto    |
| Jerry     | Kanaxo   |
| Philip    | Jose     |
| Jonh     | Abraham  |
| Michael   | Mathew   |
| Alex     | Chreketo |
| Yohan    | Soso     |
+-----+-----+
7 rows in set (0.00 sec)
```

7. Obtener todos los valores de la columna "First_name" usando el alias "Nombre de empleado"

$\Pi_{First_name \rightarrow Nombre_de_empleado}(Employee)$

`select first_name as "Nombre de empleado" from Employee;`

```
mysql> select first_name as "Nombre de empleado" from employee;
+-----+
| Nombre de empleado |
+-----+
| Bob                |
| Jerry              |
| Philip             |
| Jonh               |
| Michael            |
| Alex               |
| Yohan              |
+-----+
7 rows in set (0.00 sec)
```

8. Obtener todos los valores de la columna "Last_name" en minúsculas.

$\Pi_{lower(Last_name)}(Employee)$

`select lower(last_name) from Employee;`

```
mysql> select LOWER (last_name) from employee;
+-----+
| LOWER (last_name) |
+-----+
| kinto             |
| kanaxo            |
| jose              |
| abraham           |
| mathew            |
| chreketo          |
| soso              |
+-----+
7 rows in set (0.01 sec)
```

9. Obtener todos los valores de la columna "Last_name" en mayúsculas.

$\Pi_{upper(Last_name)}(Employee)$


```
select upper(last_name) from Employee;
```

```
mysql> select UPPER (last_name) from employee;
+-----+
| UPPER (last_name) |
+-----+
| KINTO              |
| KANAXO              |
| JOSE                |
| ABRAHAM             |
| MATHEW              |
| CHREKETO            |
| SOSO                |
+-----+
7 rows in set (0.01 sec)
```

10. Obtener los nombres únicos de la columna "Departament".

$\Pi_{\text{Departament}}(\text{Employee})$

```
select distinct Departament from Employee;
```

```
mysql> select departament from employee;
+-----+
| departament |
+-----+
| Finance     |
| IT          |
| Banking     |
| Insurance   |
| Finance     |
| IT          |
| Banking     |
+-----+
7 rows in set (0.00 sec)
```

11. Obtener los primeros 4 caracteres de todos los valores de la columna "First_name".

$\Pi_{\text{substr}(\text{First_name},1,4)}(\text{Employee})$

```
select left (first_name , 4) from Employee;
```

```
mysql> select left (first_name, 4) from employee;
+-----+
| left (first_name, 4) |
+-----+
| Bob                  |
| Jerr                  |
| Phil                 |
| Jonh                 |
| Mich                 |
| Alex                 |
| Yoha                  |
+-----+
7 rows in set (0.01 sec)
```

12. Obtener la posición de la letra 'h' en el nombre del empleado con First_name = 'John'.

$\Pi_{pos("h", First_name)}(First_name = "Jhon"(Employee))$

`select instr(first_name, 'h') from employee where first_name = 'John';`

```
mysql> select instr (first_name, 'h') from employee;
+-----+
| instr (first_name, 'h') |
+-----+
| 0 |
| 0 |
| 2 |
| 4 |
| 4 |
| 0 |
| 3 |
+-----+
7 rows in set (0.01 sec)
```

13. Obtener todos los valores de la columna "First_name" después de remover los espacios en blanco de la derecha.

$\Pi_{rtrim(First_name)}(Employee)$

`select rtrim(first_name) from employee;`

```
mysql> select rtrim (first_name) from employee;
+-----+
| rtrim (first_name) |
+-----+
| Bob |
| Jerry |
| Philip |
| Jonh |
| Michael |
| Alex |
| Yohan |
+-----+
7 rows in set (0.01 sec)
```


14. Obtener todos los valores de la columna "First_name" después de remover los espacios en blanco de la izquierda.

$\Pi_{ltrim(First_name)}(Employee)$

```
select ltrim(first_name) from employee;
```

```
mysql> select ltrim (first_name) from employee;
+-----+
| ltrim (first_name) |
+-----+
| Bob                |
| Jerry              |
| Philip             |
| Jonh               |
| Michael            |
| Alex               |
| Yohan              |
+-----+
7 rows in set (0.01 sec)
```

Conclusión

Mediante estos ejercicios prácticos, se logro reforzar la comprensión de los conceptos fundamentales del álgebra relacional y SQL. Demostramos una aplicación directa de las operaciones unarias y binarias del álgebra relacional para construir consultas SQL, demostrando cómo ambos lenguajes se complementan para la manipulación y gestión eficiente de las bases de datos.

Los ejercicios prácticos nos ayudaron a mejorar las habilidades para crear estructuras de datos, insertar registros y ejecutar consultas con funciones de manipulación de texto, sentando las bases para trabajar con bases de datos distribuidas más complejas.